fminsearch

Find minimum of unconstrained multivariable function using derivative-free method

Equation

Finds the minimum of a problem specified by

$$\min_x f(x)$$

where *f*(*x*) is a function that returns a scalar.

x is a vector or a matrix; see Matrix Arguments.

Syntax

```
x = fminsearch(fun,x0)
x = fminsearch(fun,x0,options)
x = fminsearch(problem)
[x,fval] = fminsearch(...)
[x,fval,exitflag] = fminsearch(...)
[x,fval,exitflag,output] = fminsearch(...)
```

Description

`fminsearch` attempts to find a minimum of a scalar function of several variables, starting at an initial estimate. This is generally referred to as *unconstrained nonlinear optimization*.

> **Note:** Passing Extra Parameters explains how to pass extra parameters to the objective function, if necessary.

`x = fminsearch(fun,x0)` starts at the point `x0` and returns a value `x` that is a local minimizer of the function described in `fun`. `fun` is either a function handle to a file or is an anonymous function. `x0` can be a scalar, vector, or matrix.

`x = fminsearch(fun,x0,options)` minimizes with the optimization options specified in the structure `options`. Use `optimset` to set these options.

`x = fminsearch(problem)` finds the minimum for `problem`, where `problem` is a structure described in Input Arguments.

Create the structure `problem` by exporting a problem from Optimization app, as described in Exporting Your Work.

`[x,fval] = fminsearch(...)` returns in `fval` the value of the objective function `fun` at the solution `x`.

`[x,fval,exitflag] = fminsearch(...)` returns a value `exitflag` that describes the exit condition of fminsearch.

`[x,fval,exitflag,output] = fminsearch(...)` returns a structure `output` that contains information about the optimization.

Input Arguments

Function Arguments contains general descriptions of arguments passed into `fminsearch`. This section provides function-specific details for `fun`, `options`, and `problem`:

`fun` — The function to be minimized. `fun` is a function handle for a function that accepts a vector `x` and returns a scalar `f`, the objective function evaluated at `x`. The function `fun` can be specified as a function handle for a file:

```
x = fminsearch(@myfun,x0)
```

where `myfun` is a MATLAB® function such as

```
function f = myfun(x)
f = ...            % Compute function value at x
```

`fun` can also be a function handle for an anonymous function, such as

```
x = fminsearch(@(x)norm(x)^2,x0,A,b);
```

`options` — Options provides the function-specific details for the `options` values.

`problem`:

`objective`	Objective function
`x0`	Initial point for `x`
`solver`	`'fminsearch'`
`options`	Options structure created using `optimset`

Output Arguments

Function Arguments contains general descriptions of arguments returned by `fminsearch`. This section provides function-specific details for `exitflag` and `output`:

`exitflag`	Integer identifying the reason the algorithm terminated. The following lists the values of `exitflag` and the corresponding reasons the algorithm terminated.	
	`1`	The function converged to a solution `x`.
	`0`	Number of iterations exceeded `options.MaxIter` or number of function evaluations exceeded `options.MaxFunEvals`.
	`-1`	The algorithm was terminated by the output function.
`output`	Structure containing information about the optimization. The fields of the structure are	
	`iterations`	Number of iterations
	`funcCount`	Number of function evaluations
	`algorithm`	`'Nelder-Mead simplex direct search'`
	`message`	Exit message

Options

Optimization options used by `fminsearch`. You can use `optimset` to set or change the values of these fields in the options structure `options`. See Optimization Options Reference for detailed information.

`Display`

Level of display (see Iterative Display):

- `'off'` or `'none'` displays no output.
- `'iter'` displays output at each iteration.
- `'notify'` displays output only if the function does not converge.
- `'final'` (default) displays just the final output.

`FunValCheck`

Check whether objective function values are valid. `'on'` displays an error when the objective function returns a value that is `complex` or `NaN`. The default `'off'` displays no error.

`MaxFunEvals`

Maximum number of function evaluations allowed, a positive integer. The default is `200*numberOfVariables`. See Tolerances and Stopping Criteria and Iterations and Function Counts.

`MaxIter`

Maximum number of iterations allowed, a positive integer. The default value is `200*numberOfVariables`. See Tolerances and Stopping Criteria and Iterations and Function Counts.

`OutputFcn`

Specify one or more user-defined functions that an optimization function calls at each iteration, either as a function handle or as a cell array of function handles. The default is none (`[]`). See Output Function.

`PlotFcns`

Plots various measures of progress while the algorithm executes, select from predefined plots or write your own. Pass a function handle or a cell array of function handles. The default is none (`[]`):

- `@optimplotx` plots the current point.
- `@optimplotfunccount` plots the function count.
- `@optimplotfval` plots the function value.

For information on writing a custom plot function, see Plot Functions.

`TolFun`

Termination tolerance on the function value, a positive scalar. The default is `1e-4`. See Tolerances and Stopping Criteria. Unlike other solvers, `fminsearch` stops when it satisfies *both* `TolFun` and `TolX`.

`TolX`

Termination tolerance on `x`, a positive scalar. The default value is `1e-4`. See Tolerances and Stopping Criteria. Unlike other solvers, `fminsearch` stops when it satisfies *both* `TolFun` and `TolX`.

Examples

Example 1: Minimizing Rosenbrock's Function with fminsearch

A classic test example for multidimensional minimization is the Rosenbrock banana function:

$$f(x) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2.$$

The minimum is at (1,1) and has the value 0. The traditional starting point is (-1.2,1). The anonymous function shown here defines the function and returns a function handle called banana:

```
banana = @(x)100*(x(2)-x(1)^2)^2+(1-x(1))^2;
```

Pass the function handle to fminsearch:

```
[x,fval,exitflag] = fminsearch(banana,[-1.2, 1])
```

This produces

```
x =
    1.0000    1.0000

fval =
    8.1777e-010

exitflag =
     1
```

This indicates that the minimizer was found at [1 1] with a value near zero.

Example 2

You can modify the first example by adding a parameter *a* to the second term of the banana function:

$$f(x) = 100\left(x_2 - x_1^2\right)^2 + (a - x_1)^2.$$

This changes the location of the minimum to the point [a,a^2]. To minimize this function for a specific value of a, for example a = sqrt(2), create a one-argument anonymous function that captures the value of a.

```
a = sqrt(2);
banana = @(x)100*(x(2)-x(1)^2)^2+(a-x(1))^2;
```

Then the statement

```
[x,fval,exitflag] = fminsearch(banana, [-1.2, 1], ...
   optimset('TolX',1e-8))
```

seeks the minimum [sqrt(2), 2] to an accuracy higher than the default on x. The result is

```
x =
    1.4142    2.0000

fval =
  4.2065e-018

exitflag =
     1
```

Limitations

fminsearch solves nondifferentiable problems and can often handle discontinuity, particularly if it does not occur near the solution. fminsearch might only give local solutions.

fminsearch only minimizes over the real numbers, that is, *x* must only consist of real numbers and *f*(*x*) must only return real numbers. When *x* has complex variables, they must be split into real and imaginary parts.

Notes

fminsearch is not the preferred choice for solving problems that are sums of squares, that is, of the form

$$\min_x \|f(x)\|_2^2 = \min_x \left(f_1(x)^2 + f_2(x)^2 + \ldots + f_n(x)^2\right)$$

Instead use the lsqnonlin function, which has been optimized for problems of this form.

More About

collapse all

Algorithms

`fminsearch` uses the simplex search method of [1]. This is a direct search method that does not use numerical or analytic gradients as in `fminunc`. The algorithm is described in detail in fminsearch Algorithm.

`fminsearch` is generally less efficient than `fminunc` for problems of order greater than two. However, when the problem is highly discontinuous, `fminsearch` might be more robust.

- Create Function Handle
- Anonymous Functions

References

[1] Lagarias, J. C., J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," *SIAM Journal of Optimization*, Vol. 9, Number 1, pp. 112–147, 1998.

See Also

`fminbnd` | `fminunc` | `optimset` | `optimtool`